Mshell.

Класс Mshell предназначен для кэширования результатов запросов к базам данных и HTML-разметки сгенерированных страниц. В качесве кэширующего сервера может использоваться Memcached или Redis. Для работы требуется класс \_PDO (<https://github.com/avtomon/_PDO>), расширение PHP Memcache или Redis, а так же доступ к соответствующему серверу.

Описание методов

**Класс class.mshell.php**

create

public static function create ($cache\_type = *MSHELL\_CACHE\_TYPE*,  
 $connect\_type = *MSHELL\_CONNECT\_TYPE*,  
 $hostorsock = *MSHELL\_SOCKET*,  
 $port = *MSHELL\_PORT*,  
 $ttl = *MSHELL\_TTL*,  
 $tag\_ttl = *MSHELL\_TAG\_TTL*,  
 $delay = *MSHELL\_DELAY*,  
 $solt = *MSHELL\_SOLT*)

**Описание:**

Синглтон для получения объекта управления memcached.

**Параметры:**

* *$hostorsock* - имя, ip-адрес хоста или UNIX-сокет для подключения к Memcached;
* *$dbdriver* - драйвер доступа к СУБД. На данный момент поддерживаются СУБД MySQL и PostgreSQL, допустимые значения: pgsql, mysql;
* *$port* - порт, на котором Memcached слушает подключения (0 для Unix-сокета);
* *$ttl* - время жизни элемента кэша;
* *$tag\_ttl* - время жизни тега сброса для кэша;
* *$delay* - задержка между двумя последовательными запросами на получение блокировки на запись в кэш;
* *$solt* - дополнительная соль для хэширования ключей кэша.

**Пример использования:**

$mc = MShell::create();

getInstance

public static function getInstance ()

**Описание:**

Получить уже созданный экземпляр класса.

**Параметры:** нет.

**Пример использования:**

$current\_cache = $c->getInstance

getKey

private function getKey ($query)

**Описание:**

Функция получения ключа элемента кэша.

**Параметры:**

* *$query* – текст запроса. В качестве ключа используется md5-хэш от конкатенации текста запроса и соли, заданной при создании объекта.

**Пример использования:**

$key = $this->getKey($query . *serialize*($params));

beginTransaction

public function beginTransaction ()

**Описание:**

Стартует транзакцию для исполнение в базе данных. Не выбрасывает исключение если транзакция стартуется повторно.

**Параметры:** нет.

**Пример использования:**

$cache->beginTransaction();

commit

public function commit ()

**Описание:**

Фиксирует транзакцию в базе данных. Не выбрасывает исключение если открытой транзакции нет.

**Параметры:** нет.

**Пример использования:**

$cache->commit();

rollBack

public function rollBack ()

**Описание:**

Откатывает транзакцию. Не выбрасывает исключение если нет открытой транзакции.

**Параметры:** нет.

**Пример использования:**

$cache->rollBack();

getValue

public function getValue ($query, array $params = array(), $expires = 120)

**Описание:**

Возвращает данные. Если надо лезет за ними в базу, иначе берет из кэша

**Параметры:**

* *$query* - текст запроса;
* *$params* - массив параметров запроса;
* *$expires* - время, в течение которого элемент кэша считается актульным

**Пример использования:**

$result = $cache->getValue($sql, $data, 0);

initTags

private function initTags (array $tags = array())

**Описание:**

Инициализация заданного массива тегов.

**Параметры:**

* *$tags* - заданный массив тегов.

**Пример использования:**

$this->initTags($tags);

setValue

private function setValue ($key, $value, $expires)

**Описание:**

Сохранение результата запроса в кэше.

**Параметры:**

* *$key* - ключ сохраняемых данных;
* *$value* - сохраняемые данные;
* *$expires* - сколько считать данные актуальными.

**Пример использования:**

$this->setValue($key, $value, $expires);

saveHTML

public function saveHTML($html, $url, $expire)

**Описание:**

Сохраняет разметку страницы в кэше.

**Параметры:**

* *$html* - html-разметка страницы;
* *$url* - урл по которому хостится страница;
* *$expires* - сколько считать данные актуальными.

**Пример использования:**

$cache->saveHTML($html, $\_REQUEST['page'], $source['expire']);

getHTML

public function getHTML ($url)

**Описание:**

Достать разметку страницы из кэша.

**Параметры:**

* *$url* - урл по которому хостится страница.

**Пример использования:**

$html = $cache->getHTML($\_REQUEST['page']);

delHTML

public function delHTML ($url)

**Описание:**

Удаляет разметку определенной страницы из кэша.

**Параметры:**

* *$url* - урл по которому хостится страница.

**Пример использования:**

$cache->delHTML($page\_url);

Механика работы

На вход модуля поступает запрос на выборку из базы данных или урл страницы, которую нужно достать из кэша.  
Если это запрос на выборку из базы, то модуль ищет соответствующие данные в кэше, если такие данные присутствуют и они актуальны - возвращает их. Если же данные отсутствую или "протухли" системы пытается достать эти данные из базы попутно выставив блокировку на соответствующий ключ, чтобы не возникало состояния гонки в доступе к базе. Результат выборки сохраняется в кэше на время указанное в параметре expire и отдается клиенту. Модуль поддерживает систему тегов. В текущей редакции теги представляют собой список таблиц базы. Если происходит изменение в какой-либо из таблиц, то соответствующий этой таблице тег сбрасывается и все элементы кэша, связанные с этой таблицей, становятся неактуальными. Это сделано для исключения ситуации отдачи из кэша неактуальных данных.  
Если же модуль попросили достать из кэша код страницы, то система смотрит есть ли актуальная копия разметки страницы и в противном случае просто возвращает false.